Robot Programming Practice #1

Dept. of Mech. Robotics and Energy Eng.
Dongguk University

Getting Started with mbed controller

- To use mbed controller, you need

1. PC and Internet connection
2. USB port

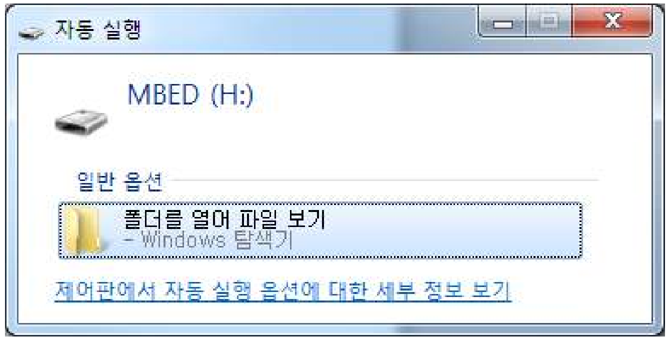
- http://www.mbed.org/
- mbed is designed for beginners with little knowledge on electronics and microcontrollers.
- The advantage of the mbed is that it doesn't require complex concepts such as flag and register.
- Basic commands such as DigitalOut are used to control ports.

Getting Started with mbed controller

- Let us connect the mbed controller to the PC as shown below.

Getting Started with mbed controller

- If you connect the mbed to the PC, then the PC will recognize the mbed as an external storage.



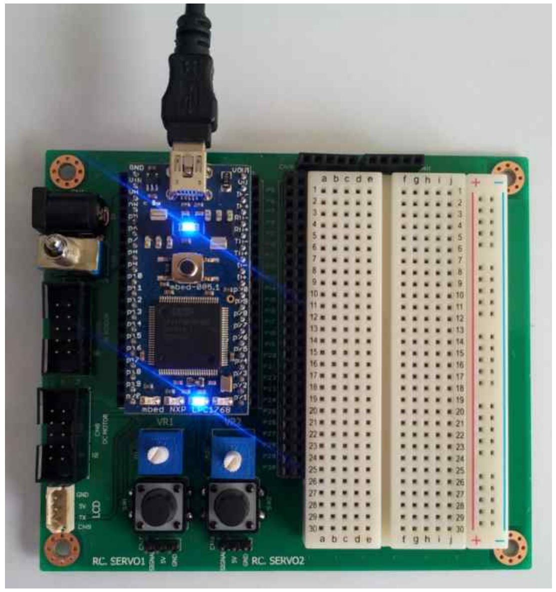
자동 실행

MBED (H:)

일반 옵션

폴더를 열어 파일 보기
- Windows 탐색기

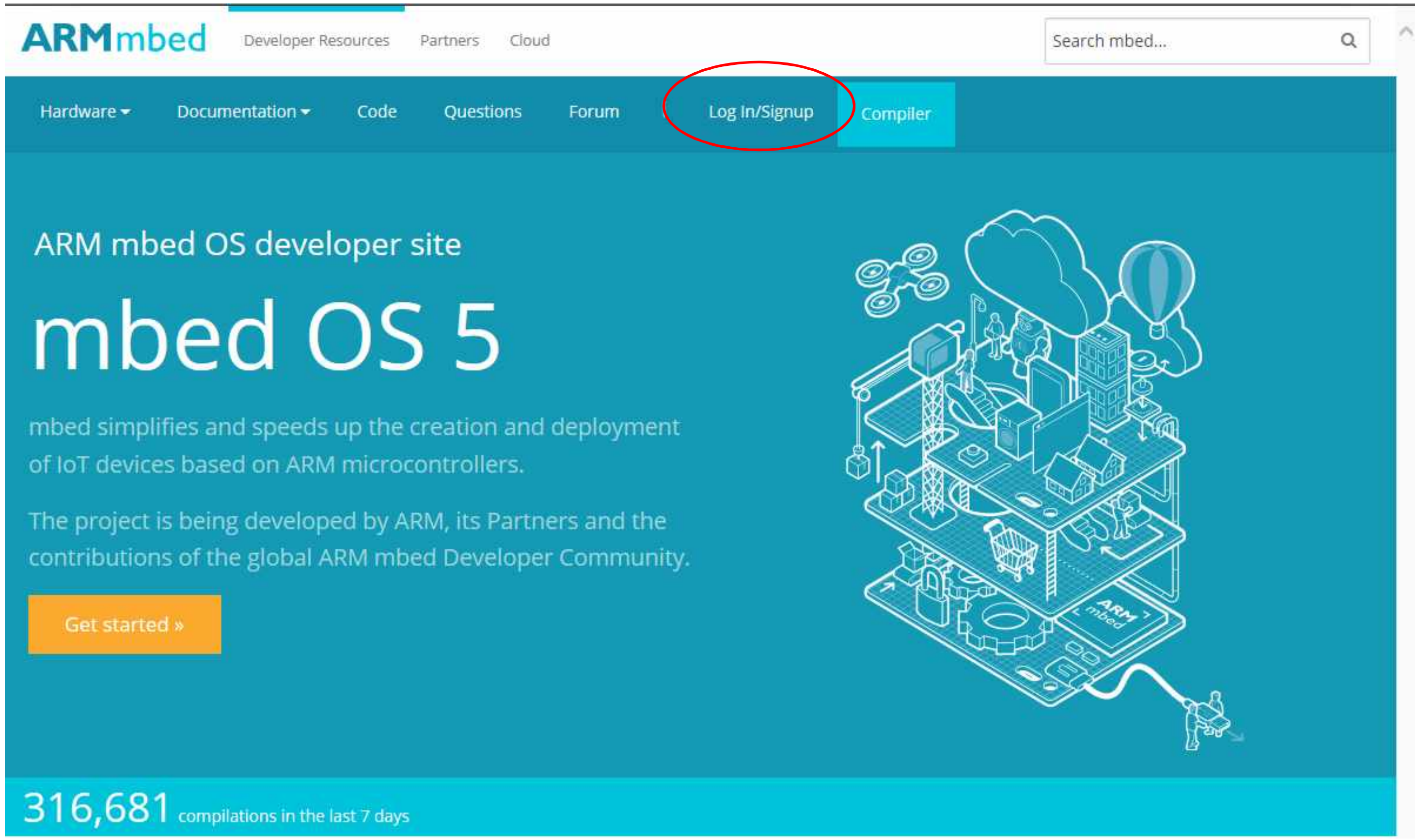
제어판에서 자동 실행 옵션에 대한 세부 정보 보기



- Close the window.

Getting Started with mbed controller

- Log into website(http://developer.mbed.org) and click Login/Signup.



ARM mbed OS developer site

mbed OS 5

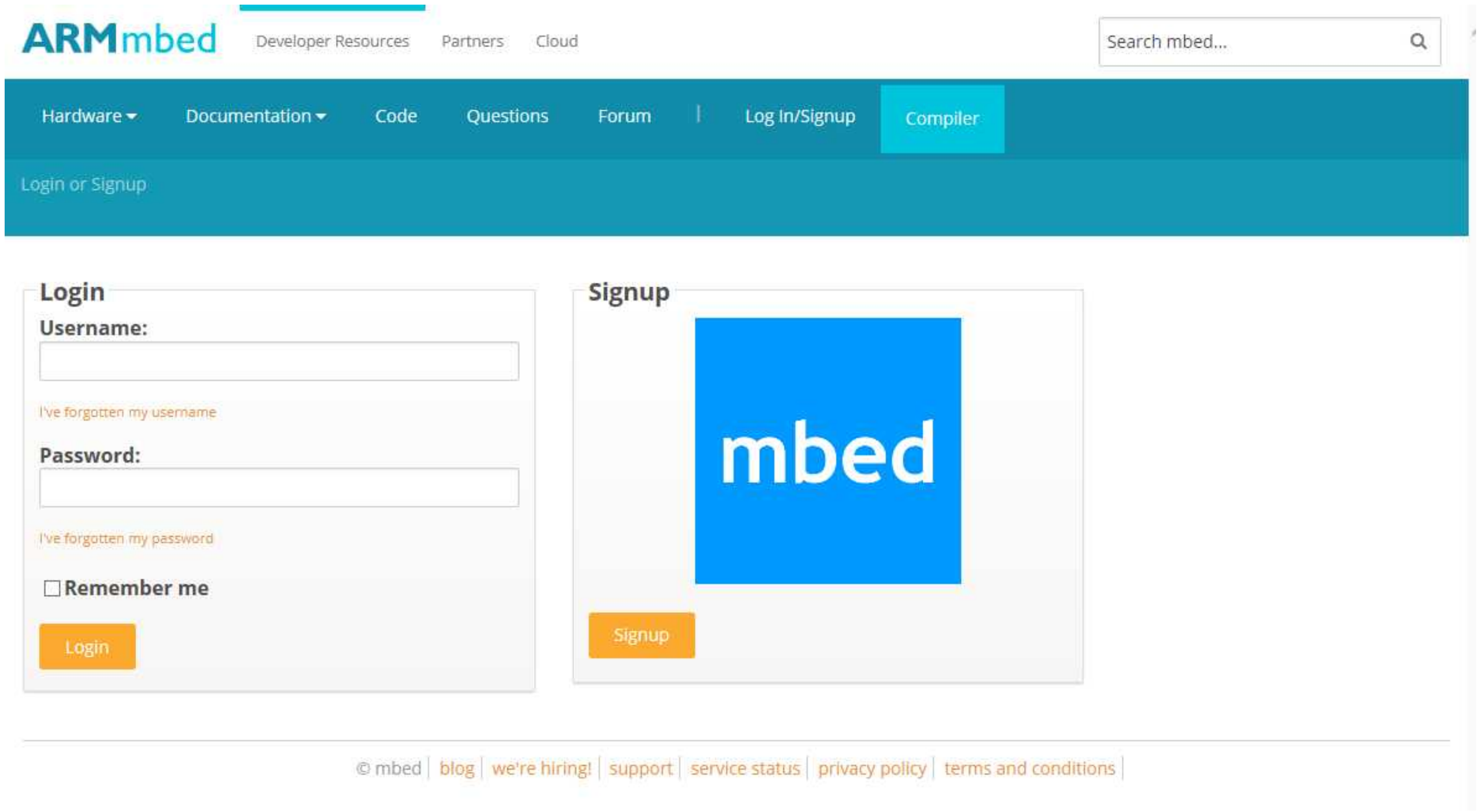
mbed simplifies and speeds up the creation and deployment of IoT devices based on ARM microcontrollers.

The project is being developed by ARM, its Partners and the contributions of the global ARM mbed Developer Community.

316,681 compilations in the last 7 days

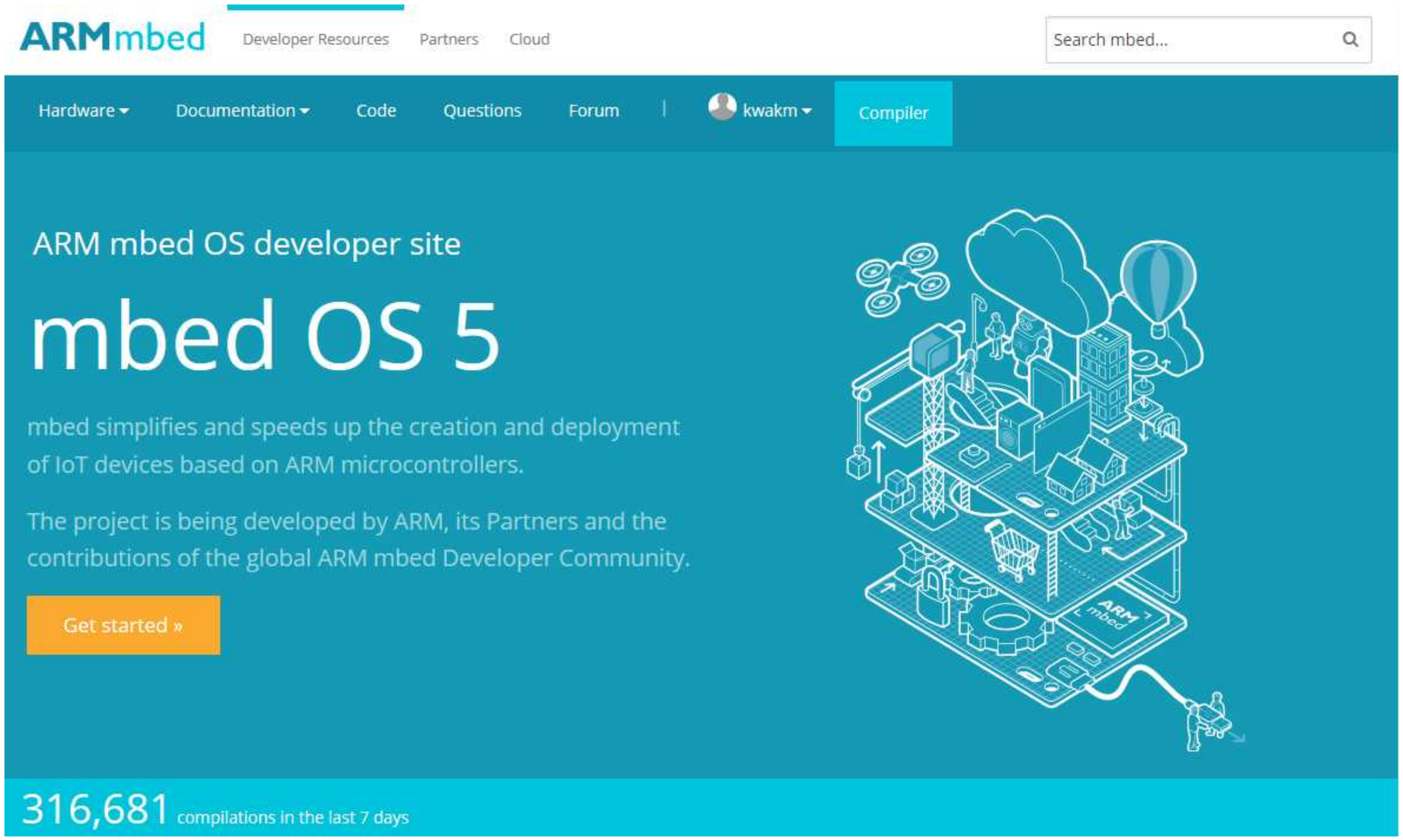
Getting Started with mbed controller

- If you click Login or signup, then you will see this. Click Signup and create account.



Getting Started with mbed controller

- After log on, you will see the following.



ARM mbed OS developer site

mbed OS 5

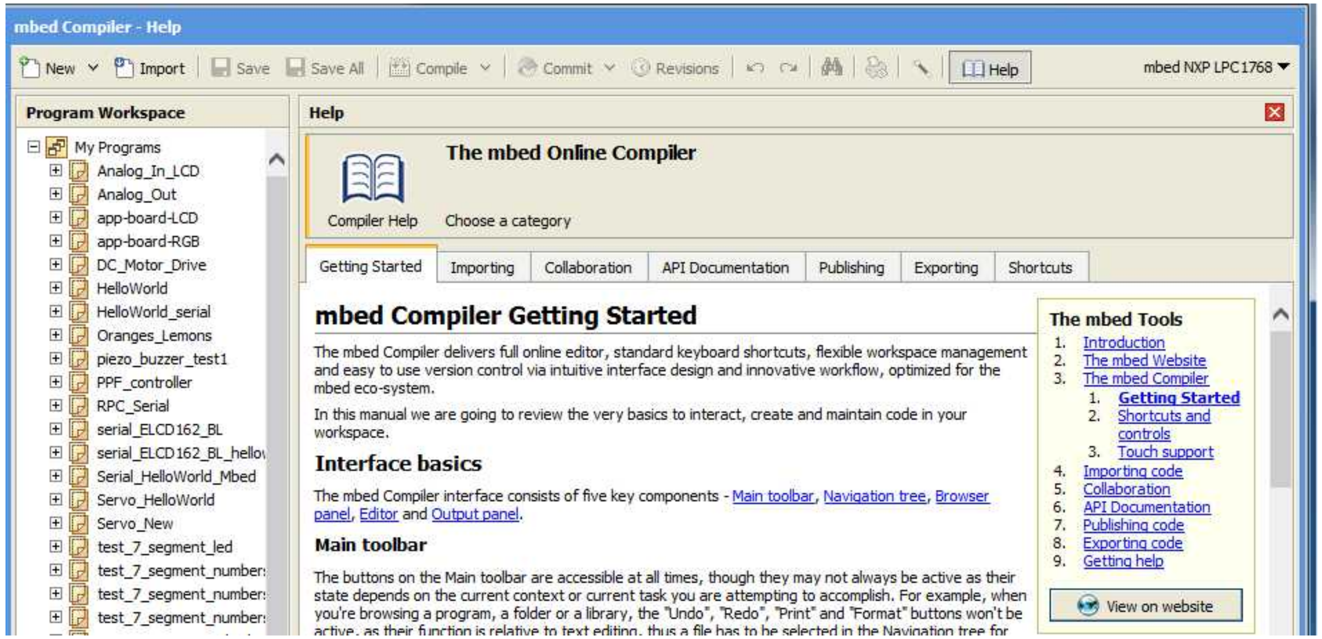
mbed simplifies and speeds up the creation and deployment of IoT devices based on ARM microcontrollers.

The project is being developed by ARM, its Partners and the contributions of the global ARM mbed Developer Community.

316,681 compilations in the last 7 days

Getting Started with mbed controller

- Click the compiler button and open the compiler window. You will see the window like below.



mbed Compiler Getting Started

The mbed Compiler delivers full online editor, standard keyboard shortcuts, flexible workspace management and easy to use version control via intuitive interface design and innovative workflow, optimized for the mbed eco-system.

In this manual we are going to review the very basics to interact, create and maintain code in your workspace.

Interface basics

The mbed Compiler interface consists of five key components - Main toolbar, Navigation tree, Browser panel, Editor and Output panel.

Main toolbar

The buttons on the Main toolbar are accessible at all times, though they may not always be active as their state depends on the current context or current task you are attempting to accomplish. For example, when you're browsing a program, a folder or a library, the "Undo", "Redo", "Print" and "Format" buttons won't be active, as their function is relative to text editing, thus a file has to be selected in the Navigation tree for

- Click New.

Getting Started with mbed controller

- Create a new program and you will the C program like below.

```
#include "mbed.h"
DigitalOut myled(LED1);

int main() {
   while(1) {
      myled = 1;
      wait(0.2);
      myled = 0;
      wait(0.2);
   }
}
```

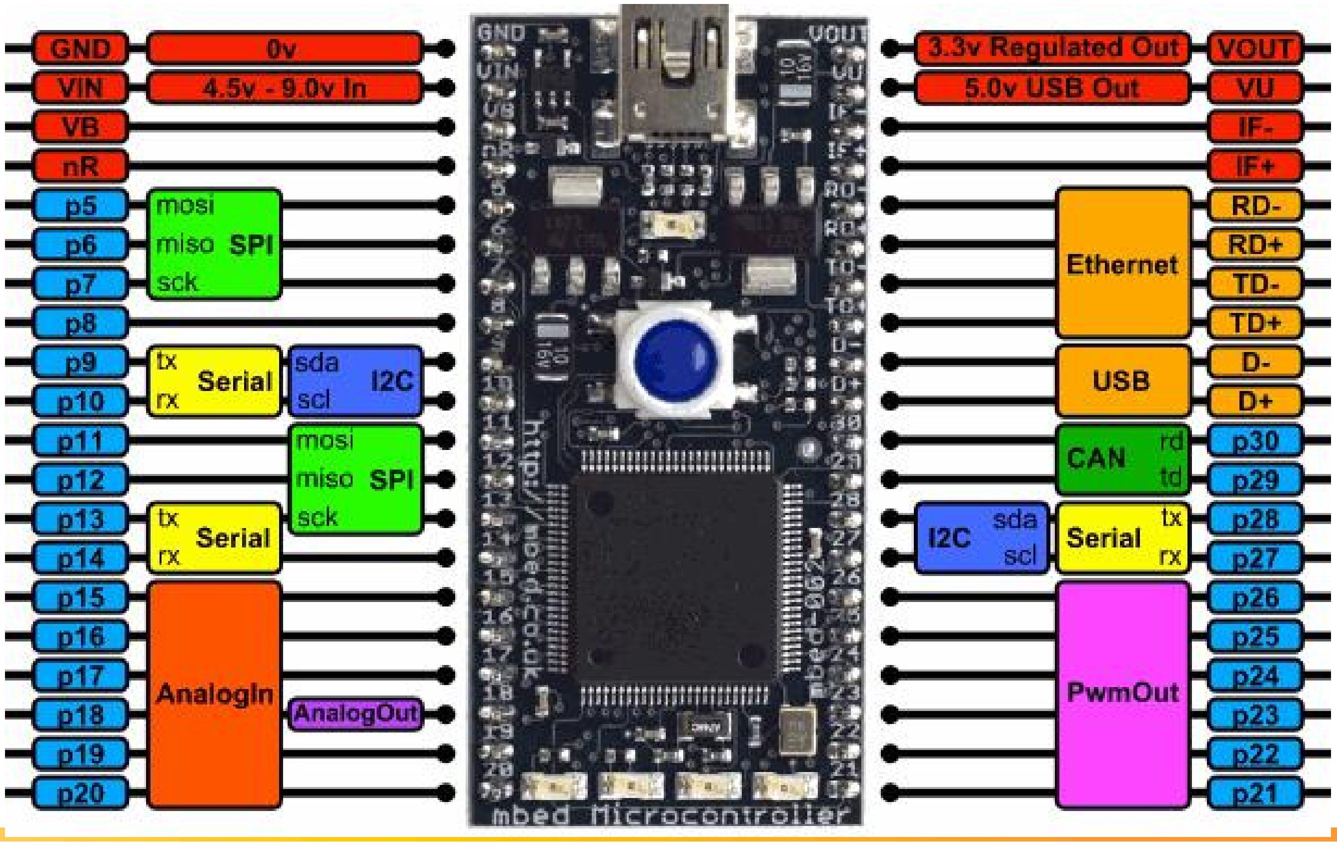
- Compile and Download the Program to the mbed.
- Downloading is saving the file to the mbed.

Getting Started with mbed controller

- What happens to the mbed?
- View the default program source code.
- Can you tell me what is going on?

mbed NXP LPC1768 Getting Started

- **Rapid Prototyping for general microcontroller applications, Ethernet, USB and 32-bit ARM® Cortex™-M3 based designs**.



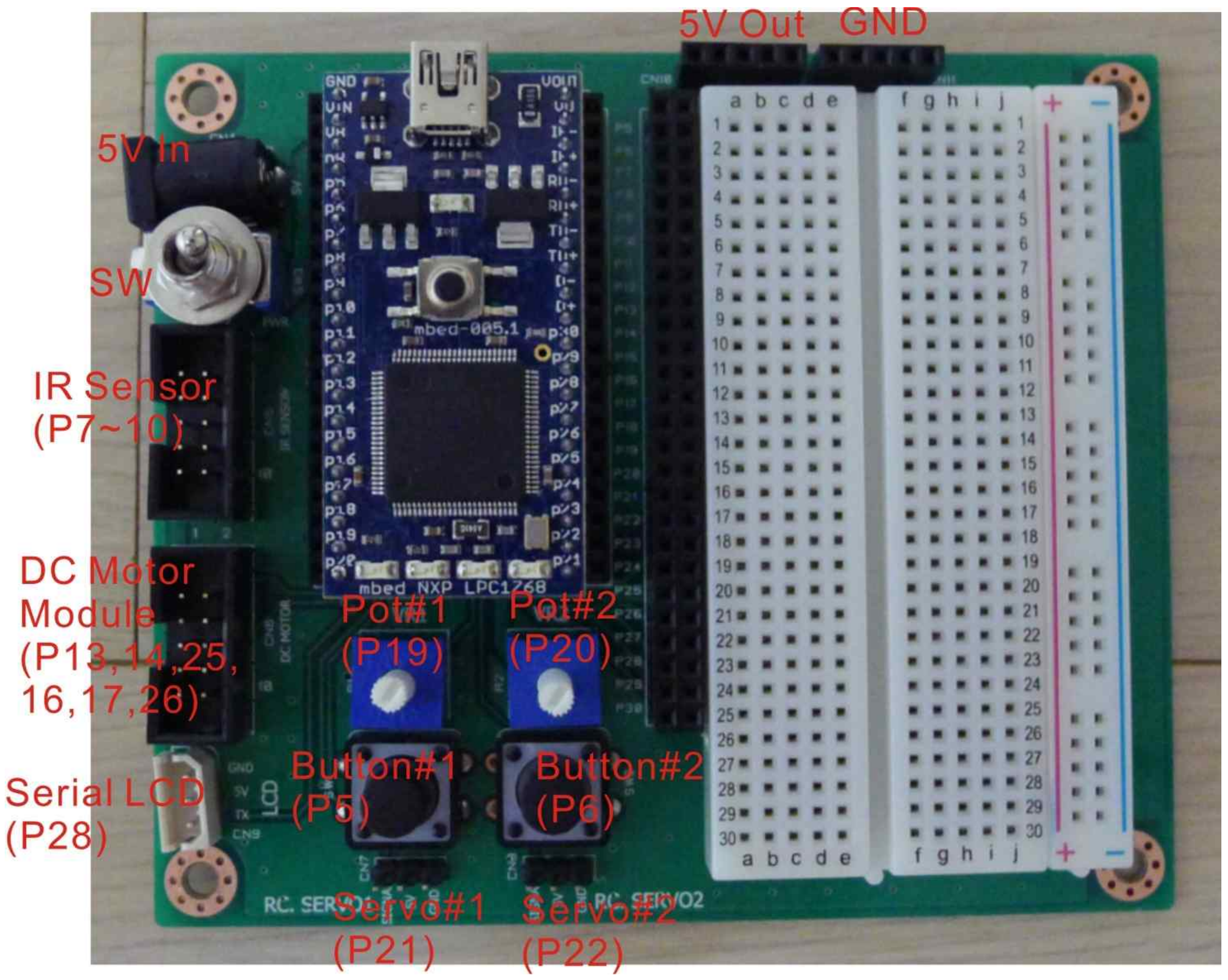
Introduction of mbed

- Ports are bi-directional I/O ports, so that we can set the direction of the port on our purpose.
- The mbed Compiler is a powerful online IDE that is free for use with hardware implementing the mbed HDK, and tightly integrated with the mbed SDK and Developer Website.
- Once the program is downloaded into the controller, it will reside on a flash memory. Hence, the program will stay on without external power supply.

mbed studyboard

- We developed the mbed studyboard for students.
- This studyboard is equipped with sensors, I/O ports and breadboard.
- Let’s take a look at the studyboard.

mbed studyboard



mbed studyboard

Module	Port	mbed pin
IR Sensor Module	Sensor 1	7
	Sensor 2	8
	Sensor 3	9
	Sensor 4	10
DC Motor Module	Dir1(DI)	13
	/Enable1(DI)	14
	PWM1	25
	Dir2(DI)	16
	/Enable2(DI)	17
	PWM2	26

module	port	mbed pin
Serial LCD	TX	28
VR1	AI	19
VR2	AI	20
Button SW1	DI	5
Button SW2	DI	6
RC Servo 1	PWM	21
RC Servo 2	PWM	22

* If there are no connections to IR sensor module, DC motor module, serial LCD, RC servos, then the corresponding ports can be used freely. However, pins 5, 6, 19 and 20 shouldn't be used in any case.

Your first C program for the mbed

- Take a look at your first main.c program again.

```
#include "mbed.h"
DigitalOut myled(LED1);

int main() {
   while(1) {
      myled = 1;
      wait(0.2);
      myled = 0;
      wait(0.2);
   }
}
```

- There are dedicated C functions and mbed functions.
- Let us study them.

A Closer Look

- Take a closer look at each line in the sample program.

```
#include "mbed.h"
```

- This command tells C to include the file mbed.h in the compilation. This file contains information needed by the program to ensure the correct operation of C for the mbed.

```
DigitalOut myled(LED1);
```

- The variable myled is defined as DigitalOut and its output pin is LED1(LED1 is a built-in LED on the mbed).

A Closer Look

```
int main() {

}
```

- main() specifies the name of a function. All C programs begin execution by calling the main() function. int implies the output of the main function is a integer value. The function should start with { and end with }.

```
while(1) {

}
```

- while (condition) iterates the content of { …} when condition is true(logic 1 is true).

A Closer Look

```
myled = 1;

myled = 0;
```

- “myled =1” implies that LED1 is on.
- “myled = 0” implies that the LED1 is off.
- 1 implies 3.3V and 0 implies 0V. We will study this later.

```
wait(0.2);
```

- wait(0.2) implies that the program waits for 0.2 s.

Introduction to digital terminology

- The mbed uses a power rail of 3.3 Volts – 0 Volts indicates ‘off’ and 3.3 Volts indicates ‘on’.
- A number of terms are used interchangeably to describe on and off in digital systems:

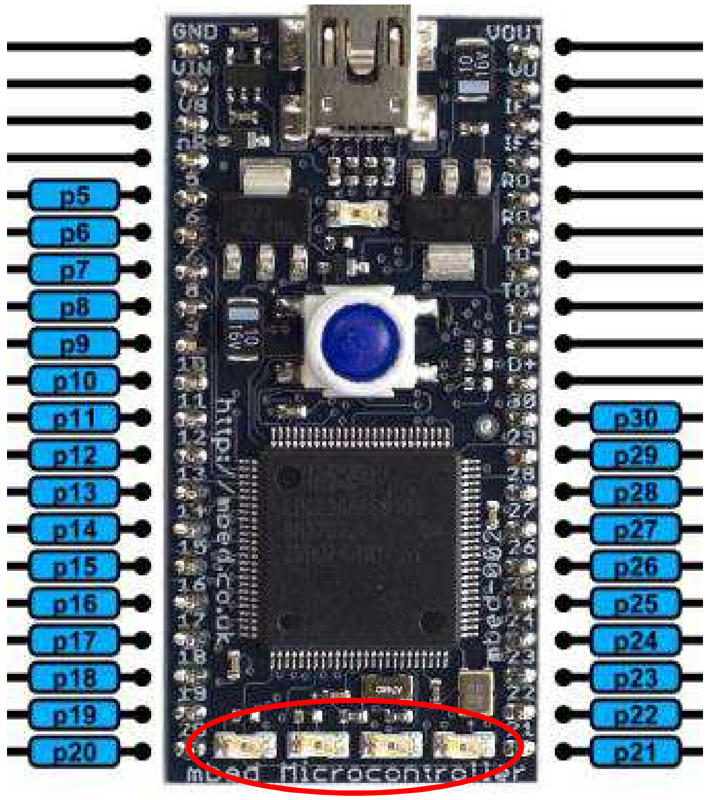
0V	3.3V
Open	Closed
Off	On
Low	High
Clear	Set
logic 0	logic 1
False	True

- Note: terms ‘logic 0’ and ‘logic 1’ may be simply referred to as ‘0’ and ‘1’.

Digital outputs on the mbed

- On the mbed, the four on-board LEDs are digital outputs which have been specially configured to operate with no extra wires or connections needed.

- The mbed also has 26 digital IO pins(pins 5-30) which can be configured as inputs or outputs.



Digital outputs on the mbed

- The digital IO pins are configured by defining them at the start of the program code.
- Each digital IO is given a name and associated pin, for example:

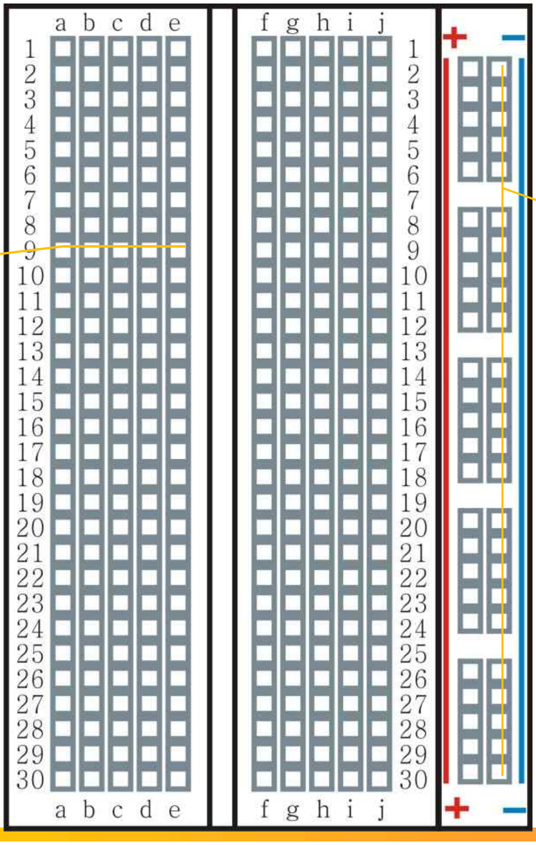
```
DigitalOut myname1(p13);
DigitalOut myname2(p14);
DigitalOut myname3(p15);
```

- The DigitalOut interface can be used to set the state of the output pin, and also read back the current output state.
- Set the DigitalOut to 0 to turn it off, or 1 to turn it on.

Breadboard

- A **breadboard** (**protoboard**) is a construction base for prototyping of electronics. The term is commonly used to refer to **solderless breadboard** (**plugboard**).

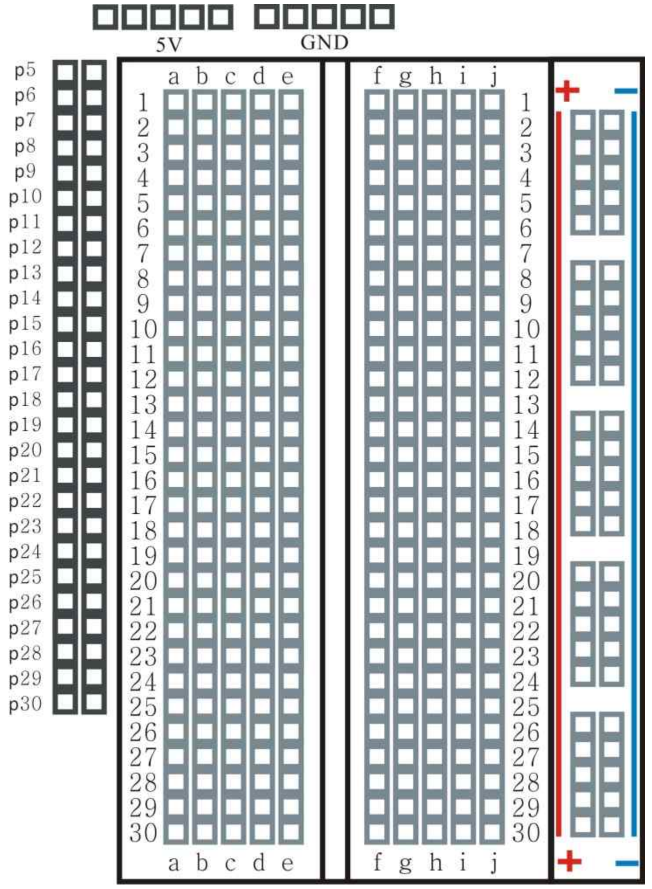
Holes of each row are connected.



+, - vertial holes of each column are connected.

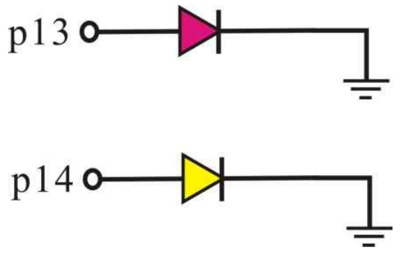
mbed studyboard

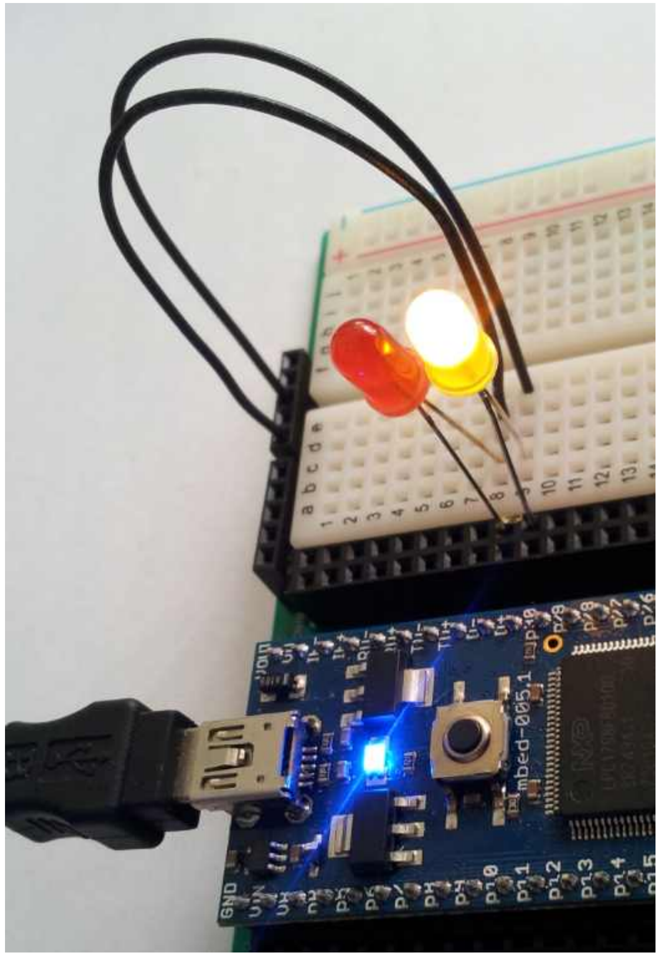
- mbed studyboard has mbed ports right next to the breadboard.



Connecting LEDs to mbed

- Connect a red LED to pin 13 and a yellow LED to pin14.





- Remember to attach the positive side of the led (the side with the longer leg) to the mbed pins. The negative side should be connected to ground. Use wires.

Connecting LEDs to mbed

- Create a new program for the external LEDs. Modify the default main.cpp code to become the following:

```
#include "mbed.h"
DigitalOut redled(p13);
DigitalOut yellowled(p14);
int main() {
    while(1) {
        redled = 1;
        yellowled = 0;
        wait(0.2);
        redled = 0;
        yellowled = 1;
        wait(0.2);
    }
}
```

- Compile, download and run the code on the mbed.

Connecting LEDs to mbed

- Look at the example program and identify the key C programming elements as follows:
  - The mbed.h library file is linked to by the `#include’ statement.
  - DigitalOut objects are defined with a name and a chosen mbed pin.
  - The main program function exists inside `int main() { ... program... }‘.
  - An infinite loop is implemented by the while(1) statement, so that the program continuously loops forever, allowing the led to flash continuously.
  - Digital outputs are controlled simply by setting the relevant objects equal to 0 or 1.
  - The mbed wait() function is used for timing control.

Push Button

- Computer, mouse, calculator, microwave oven, remote controller, game player, and cellular phone have push buttons.
- We can make a micro-controller act under the command input by a push button.
- The open- and closed-condition of the push button can be sensed by the micro-controller using the digital I/O ports.

Digital inputs on the mbed

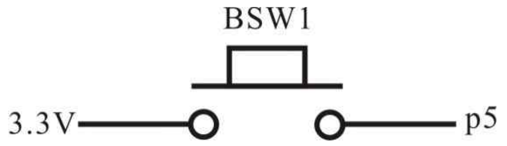
- Digital inputs values can be read.
- As with digital outputs, the same 26 pins (pins 5-30) can be configured as digital inputs, as follows:

```
DigitalIn button1(p5);
DigitalIn button2(p6);
DigitalIn mybutton3(p7);
```

- The DigitalIn Interface determines the current logical state of the chosen input pin, e.g. logic '0' or logic '1'.
- Zero volts on a digital input pin returns a logical 0, whereas 3.3 Volts returns a logic 1.

Connecting switches to the mbed

- Let us a mechanical switch mounted on the SB.
- Button SW #1 is already connected to pin 5.
- The other side of the button switch is connected to 3.3V.



- A digital input pin is set to either high (‘1’) or low (‘0’) by connecting it to switch between the 3.3V and GND rails.

Implementing a digital switch input

- Create a new program for the LED switch project. Modify the default main.cpp code as shown.

- When two forward slash symbols (//) are used, the compiler ignores any proceeding text, so we can use this to write useful comments.

```
#include "mbed.h"
DigitalOut redled(p13);
DigitalOut yellowled(p14);
DigitalIn sw1(p5);

int main() {
    while(1) {

        if(sw1==1) {
           redled = 1;      // red led on
           yellowled = 0;   // yellow led off
}
        else if (sw1==0) {
           redled = 0;      // red led off
           yellowled = 1;   // yellow led on
}
    }
}
```

Implementing a digital switch input

- Modify a program. Red or yellow LED will flash by the SW.

```
#include "mbed.h"
DigitalOut redled(p13);
DigitalOut yellowled(p14);
DigitalIn sw1(p5);

int main() {
    while(1) {

        if(sw1==1) {
           yellowled = 0;  // yellow led off
           redled = 1;     // flash red led
           wait(0.2);
           redled = 0;
           wait(0.2);
        }
        else if (sw1==0) {
           redled = 0;     // red led ff
           yellowled = 1;  // flash yellow led
           wait(0.2);
           yellowled = 0;
           wait(0.2);
        }
    }
}
```

Implementing a digital switch input

- Look at the code, the “if (sw1==1)…” statement allows the code to operate in two different ways, depending on the value of the digital input (i.e. the mechanical switch position).

- If the switch gives a value of 1, the yellow LED is set to zero (off) and the red LED is programmed to flash. If the digital input is low (0), we see the roles of the LEDs reversed.

A Closer Look

```
if(sw1==1) {
      .
      .
      .
}
else if (sw1==0) {
      .
      .
      .
}
```

- If (condition) : if condition is true, then execute the content of { }.
- else : if condition is not true, then the execute this.